COMPUTER ORGANIZATION & ASSEMBLY LANGUAGE
ASSIGNMENT#2 (Spring 2024)

ID: 22i-1552 NAME: Syed Apham Ahmed SECTION: CY-B

Read the Instructions Carefully

DUE DATE: 30th March, 2024 11:59 PM

Print this assignment and complete it on the printed sheet only. No extra sheet is allowed.
Scan/Camscan the solution and upload it on Google Classroom.
You can verify the answers using MASM debugging mode.
Cheating in any case if found will be marked as ZERO in whole assignment category.
Perform all the steps in the "calculation" box, only filling the answer will not get any credit.

1. **Update Flags after executing following code?**

NOTE: AND performs a Boolean(bitwise) **AND** operation between each pair of matching bits in two operands and place result in the destination. **AND** instruction always clear overflow and carry flags. It modifies Sign, Zero and Parity flags.

```
mov al,0AEH
and al,246
```

1010 1110
1111 0110
1010 0110

Flags	Sign	1
	Zero	0
	Carry	0
	Overflow	0
	Parity	1
	Auxiliary	0

] cleared
] un-changed

2. **Update memory after executing code given below**

CODE

```
.data
      ary db 26 dup(?)
      arysize= $-ary
      ary_copy db arysize dup(0)
      endmem db 1
.code

main PROC
      mov esi,OFFSET ary
      mov edi,OFFSET ary_copy
      mov ax,000FFh
      mov ecx,arysize
      and al,061h
      mov bl,al
      and bl,0CFH       ;masking
      L1:
            mov [esi],al
            mov [edi],bl
            inc esi
            inc edi
            inc al
```

ax = 00FF
FF and 61 ⇒ 61
al, 61
bl, 61
61, 41

```
        inc bl
    LOOP L1
        INVOKE ExitProcess,0
main ENDP
END main
```

MEMORY

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4000	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	70
4010	71	72	73	74	75	76	77	78	79	7A	41	42	43	44	45	46
4020	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56
	57	58	59	5A	01											

3. Update Flags after executing following code

NOTE: OR performs a Boolean(bitwise) **OR** operation between each pair of matching bits in two operands and place result in the destination. **OR** instruction always clear overflow and carry flags. It modifies Sign, Zero and Parity flags.

```
mov al,11100011b
OR al,00000100b ; setting 3rd bit
```

```
111 00011
000 00100
---------
111 00111
```

Flags	**Sign**	1
	Zero	0
	Carry	0
	Overflow	0
	Parity	1
	Auxiliary	0

4. Update memory after executing code given below

CODE

```
.data
      ary db 26 dup(?)
      arysize= $-ary
      ary_copy db arysize dup(0)
      endmem db 1

.code
      mov esi,OFFSET ary
      mov edi,OFFSET ary_copy
      mov ax,0041h
      mov ecx,arysize
      mov bl,al
      OR bl,00100000b
      L1:
            mov [esi],al
            mov [edi],bl
            inc esi
            inc edi
            inc al
            inc bl
      LOOP L1
```

memory filled in order of allocated arr then ary_copy then endmem

ax = 0041h

bl = 41h

bl = 0110 0001 = 61h

MEMORY

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4000	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50
4010	51	52	53	54	55	56	57	58	59	5A	61	62	63	64	65	66
4020	67	68	69	6A	6B	6C	6D	6E	6F	70	71	72	73	74	75	76

77 78 79 7A 01

5. Update register after each line of code and update Flags after executing the following code

```
mov ax,0A593H
XOR ax,-1
XOR ax,0
XOR ax,-1
XOR ax,ax

mov ax, 05A37H
XOR al,0

XOR ah,1
XOR al,ah
```

Flags	
Sign	0
Zero	0
Carry	0
Overflow	0
Parity	1
Auxiliary	0

CC = 0110 1100

AX	A5	93
AX	5A	6C
AX	5A	6C
AX	A5	93
AX	00	00
AX	5A	37
Al	3	7
AH	6	B
AL	6	C

1111 1111 1111

0101

5A XOR 1

5B

37 XOR 5B

6. Write a program that finds parity of number given below? HINT: Use XOR and LOOP to find parity

```
.data
    parity DQ 0A1B2C3D4E5F67890H
```

0000 1100
0000 0000
0000 0000

0000 1011
0000 1011
0000 0000

Answer:

```
dosseg
.model small
.stack 200h
.data
parity DQ 0A1B2C3D4E5F67890H
even db 10,13,'Even parity$'
odd db 10,13,'Odd parity$'
.code
main proc
mov ax,@data
mov ds,ax
mov al, byte ptr parity
XOR al,0
jnp odd_p
mov dx, offset even
jmp e
odd_p:
mov dx, offset odd
e:
mov ah,09h
int 21h
mov ah,4ch
int 21h
main endp
end main
```

7. **Update flags after arithmetic instruction? Also state which of the following jumps will taken or not taken**

ax = FF FE FF FE + FC 70 ① 1111 1100 0111 0110

ax = FC6E

```
.code
        mov ax,0FFFEh
        add ax,0FC70h
        jc l1
   L1: jz L2
   L2: jo L3
   L3: js L4
   L4: jp L5
   L5:
```

	TAKEN	NOT TAKEN
Jc	✓	
Jz		✓
Jo		✓
Js	✓	
jp		✓

Flags	Sign	1
	Zero	0
	Carry	1
	Overflow	0
	Parity	0

Calculation

```
1111 1111 1111 1110
+ 1111 1100 0111 0000
① 1111 1100 0110 1110
```

8. **Update flags after arithmetic instruction? Also state which of the following jumps will taken or not taken**

```
.code
        mov ax,07B1Ah
        sub ax,0CEEBh
        jc l1
   L1: jz L2
   L2: jo L3
   L3: js L4
   L4: jp L5
   L5:
```

	TAKEN	NOT TAKEN
Jc	✓	
Jz		✓
Jo	✓	
Js	✓	
jp		✓

Flags	Sign	1
	Zero	0
	Carry	1
	Overflow	1
	Parity	0

Calculation

```
0111 1011 0001 1010
0011 0001 0001 0101
1010 1100 0010 1111
```

7B1A - CEEB ax = 7B1A

```
0011 0001 0001 0100
+                 1
0011 0001 0001 0101
```

ax = AC2F

9. Update flags after arithmetic instruction? Also state which of the following jumps will taken or not taken

```
.code
        mov cx,0FBCDh
        add cx,0E4AAh
        jnc l1
   L1:  jnz L2
   L2:  jo L3
   L3:  js L4
   L4:  jnp L5
```

	TAKEN	NOT TAKEN
Jnc		✓
Jnz	✓	
Jo		✓
Js	✓	
jnp		✓

Flags		
	Sign	1
	Zero	0
	Carry	1
	Overflow	0
	Parity	1
	Auxiliary	1

Calculation

```
 1111 1011 1100 1101
+1110 0100 1010 1010
-------------------
①1110 0000 0111 0111
```

1E077

10. Update flags after arithmetic instruction? Also state which of the following jumps will taken or not taken

```
        mov bx,0FABDh
        add bx,0684Ah
        jc l1
   L1:  jz L2
   L2:  jno L3
   L3:  jns L4
   L4:  jp L5
   L5:  mov ah,04ch
```

	TAKEN	NOT TAKEN
Jc	✓	
Jz		✓
Jno	✓	
Jns	✓	
jp		✓

Flags		
	Sign	0
	Zero	0
	Carry	1
	Overflow	0
	Parity	0
	Auxiliary	1

Calculation

```
 1111 1010 1011 1101
+0110 1000 0100 1010
-------------------
①0 1100 0100 0001 0111
```

11. Update value of ax and cx registers after every iteration. Update any changes to the done to flag

```
    mov ecx,5
    mov ax,1
L1:
    inc ax
    dec ecx
jcxz end_loop
jmp L1
end_loop:
```

	1	2	3	4	5	6
CX	05	04	03	02	01	00
AX	01	02	03	04	05	06

ECX = 0

Flags			Calculation
	Sign	0	ECX is = 0 in the end.
	Zero	1	
	Carry	0	
	Overflow	0	
	Parity	1	
	Auxiliary	0	

12. Fill flag after every CMP instruction

al = 7F

al - (-128)

FF

1000 0000

0111 1111

+ ———

1111 1111

```
mov al, +127
CMP al,-128
ja IsAbove
jg IsGreater
```

	TAKEN	NOT TAKEN
Ja		✓
Jg	✓	

Flags			Calculation
	Sign	1	0111 1111
	Zero	0	+ 1000 0000
	Carry	0	———
	Overflow	0	1111 1111
	Parity	1	255 > 128
	Auxiliary	0	

```
mov dx,-1
CMP dx,0

jnl L5
jnle L5
jl L1
```

	TAKEN	NOT TAKEN
Jnl		✓
Jnle		✓
JL	✓	

Flags			Calculation
	Sign	1	1111 1111 1111 1111
	Zero	0	- 0
	Carry	0	———
	Overflow	0	1111 1111 1111 1111
	Parity	1	
	Auxiliary	0	

dx - FF FF

dx: FF FF

32 - -35

67


```
mov bx,+32
cmp bx, -35

jng L5
jnge L5
jge L1
```

	TAKEN	NOT TAKEN
JNG		✓
JNGE		✓
JGE	✓	

Flags			Calculation
	Sign	0	0010 0000
	Zero	0	0010 0011
	Carry	1	0100 0001
	Overflow	0	
	Parity	0	
	Auxiliary	0	0100 0011

67

```
mov cx,0
cmp cx,0

jg L5
jnl L1
jle L2
```

	TAKEN	NOT TAKEN
JG		✓
JNL	✓	
JLE	✓	

Flags			Calculation
	Sign	0	0000 0000
	Zero	1	0000 0000
	Carry	0	
	Overflow	0	0 0000 0000
	Parity	1	
	Auxiliary	0	

```
mov cx,0
cmp cx,0

jl L5
jng L1
jge L2
```

	TAKEN	NOT TAKEN
JL		✓
JNG	✓	
JGE	✓	

Flags			Calculation
	Sign	0	0 0000000 0
	Zero	1	0000 0000
	Carry	0	
	Overflow	0	[illegible]